



Carnegie Mellon  
Software Engineering Institute

# An Introduction to Software Engineering Practices Using Model-Based Verification

David P. Gluch  
Jared Brockway

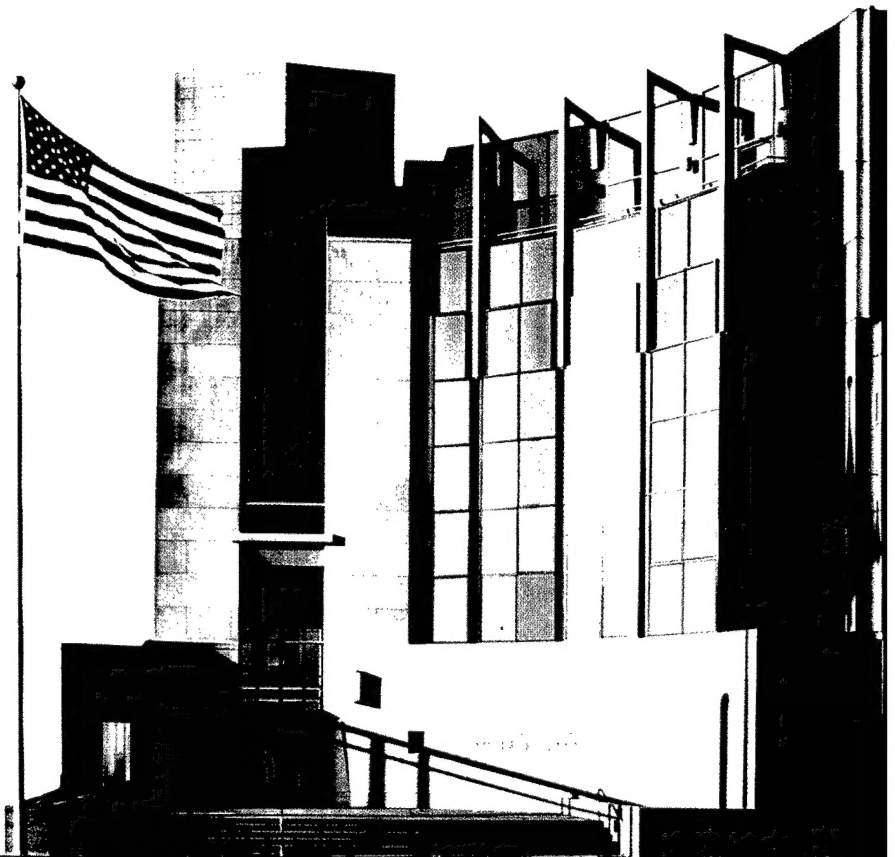
*April 1999*

19990728 002

TECHNICAL REPORT  
CMU/SEI-99-TR-005  
ESC-TR-99-005

**DISTRIBUTION STATEMENT A**  
Approved for Public Release  
Distribution Unlimited

DTIC QUALITY INSPECTED 4



Carnegie Mellon University does not discriminate and Carnegie Mellon University is required not to discriminate in admission, employment, or administration of its programs or activities on the basis of race, color, national origin, sex or handicap in violation of Title VI of the Civil Rights Act of 1964, Title IX of the Educational Amendments of 1972 and Section 504 of the Rehabilitation Act of 1973 or other federal, state, or local laws or executive orders.

In addition, Carnegie Mellon University does not discriminate in admission, employment or administration of its programs on the basis of religion, creed, ancestry, belief, age, veteran status, sexual orientation or in violation of federal, state, or local laws or executive orders. However, in the judgment of the Carnegie Mellon Human Relations Commission, the Department of Defense policy of "Don't ask, don't tell, don't pursue" excludes openly gay, lesbian and bisexual students from receiving ROTC scholarships or serving in the military. Nevertheless, all ROTC classes at Carnegie Mellon University are available to all students.

Inquiries concerning application of these statements should be directed to the Provost, Carnegie Mellon University, 5000 Forbes Avenue, Pittsburgh, PA 15213, telephone (412) 268-6684 or the Vice President for Enrollment, Carnegie Mellon University, 5000 Forbes Avenue, Pittsburgh, PA 15213, telephone (412) 268-2056.

Obtain general information about Carnegie Mellon University by calling (412) 268-2000.



**Carnegie Mellon**  
**Software Engineering Institute**  
Pittsburgh, PA 15213-3890

---

# **An Introduction to Software Engineering Practices Using Model-Based Verification**

CMU/SEI-99-TR-005  
ESC-TR-99-005

David P. Gluch  
Jared Brockway

*April 1999*

**Dependable Systems Upgrade**

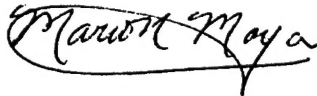
Unlimited distribution subject to the copyright.

This report was prepared for the

SEI Joint Program Office  
HQ ESC/DIB  
5 Eglin Street  
Hanscom AFB, MA 01731-2116

The ideas and findings in this report should not be construed as an official DoD position. It is published in the interest of scientific and technical information exchange.

FOR THE COMMANDER



Mario Moya, Maj, USAF  
SEI Joint Program Office

This work is sponsored by the U.S. Department of Defense. The Software Engineering Institute is a federally funded research and development center sponsored by the U.S. Department of Defense.

Copyright 1999 by Carnegie Mellon University.

#### NO WARRANTY

THIS CARNEGIE MELLON UNIVERSITY AND SOFTWARE ENGINEERING INSTITUTE MATERIAL IS FURNISHED ON AN "AS-IS" BASIS. CARNEGIE MELLON UNIVERSITY MAKES NO WARRANTIES OF ANY KIND, EITHER EXPRESSED OR IMPLIED, AS TO ANY MATTER INCLUDING, BUT NOT LIMITED TO, WARRANTY OF FITNESS FOR PURPOSE OR MERCHANTABILITY, EXCLUSIVITY, OR RESULTS OBTAINED FROM USE OF THE MATERIAL. CARNEGIE MELLON UNIVERSITY DOES NOT MAKE ANY WARRANTY OF ANY KIND WITH RESPECT TO FREEDOM FROM PATENT, TRADEMARK, OR COPYRIGHT INFRINGEMENT.

Use of any trademarks in this report is not intended in any way to infringe on the rights of the trademark holder.

Internal use. Permission to reproduce this document and to prepare derivative works from this document for internal use is granted, provided the copyright and "No Warranty" statements are included with all reproductions and derivative works.

External use. Requests for permission to reproduce this document or prepare derivative works of this document for external and commercial use should be addressed to the SEI Licensing Agent.

This work was created in the performance of Federal Government Contract Number F19628-95-C-0003 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center. The Government of the United States has a royalty-free government-purpose license to use, duplicate, or disclose the work, in whole or in part and in any manner, and to have or permit others to do so, for government purposes pursuant to the copyright license under the clause at 52.227-7013.

This document is available through Asset Source for Software Engineering Technology (ASSET): 1350 Earl L. Core Road; PO Box 3305; Morgantown, West Virginia 26505 / Phone: (304) 284-9000 or toll-free in the U.S. 1-800-547-8306 / FAX: (304) 284-9001 World Wide Web: <http://www.asset.com> / e-mail: [sei@asset.com](mailto:sei@asset.com)

Copies of this document are available through the National Technical Information Service (NTIS). For information on ordering, please contact NTIS directly: National Technical Information Service, U.S. Department of Commerce, Springfield, VA 22161. Phone: (703) 487-4600.

This document is also available through the Defense Technical Information Center (DTIC). DTIC provides access to and transfer of scientific and technical information for DoD personnel, DoD contractors and potential contractors, and other U.S. Government agency personnel and their contractors. To obtain a copy, please contact DTIC directly: Defense Technical Information Center / Attn: BRR / 8725 John J. Kingman Road / Suite 0944 / Ft. Belvoir, VA 22060-6218 / Phone: (703) 767-8274 or toll-free in the U.S.: 1-800 225-3842.

---

# Table of Contents

|  |            |
|--|------------|
| <b>Acknowledgments</b>                                     | <b>v</b>   |
| <b>Abstract</b>  | <b>vii</b> |
| <b>1 Introduction</b>                                      | <b>1</b>   |
| <b>2 Using Models for Verification</b>                     | <b>3</b>   |
| 2.1 Formal Models  | 3          |
| 2.2 Essential Models                                       | 4          |
| <b>3 Model-Based Verification Paradigm</b>                 | <b>7</b>   |
| 3.1 Building Essential Models                              | 8          |
| 3.1.1 Scope  | 9          |
| 3.1.2 Perspective  | 9          |
| 3.1.3 Formalism  | 9          |
| 3.1.4 Abstraction  | 10         |
| 3.2 Analyzing Models (Model Checking)                      | 10         |
| 3.3 Compiling Defects                                      | 12         |
| 3.4 Modifying and Extending Models                         | 12         |
| <b>4 Model-Based Verification in Peer Review Processes</b> | <b>13</b>  |
| 4.1 Reviews and Formalism                                  | 13         |
| 4.2 Integration of Processes                               | 13         |
| 4.2.1 Planning   | 15         |
| 4.2.1.1 Methods of Partitioning                            | 16         |
| 4.2.2 Coordination   | 17         |
| 4.2.2.1 Impact on Roles                                    | 17         |
| 4.2.2.2 Outcomes   | 17         |
| 4.2.3 Preparation  | 18         |
| 4.2.3.1 Impact on Roles                                    | 19         |
| 4.2.3.2 Outcomes   | 19         |
| 4.2.4 Meeting  | 19         |
| 4.2.4.1 Coordination                                       | 19         |
| 4.2.4.2 Modeler Role                                       | 20         |
| 4.2.4.3 Outcomes   | 20         |

|          |                                    |           |
|----------|------------------------------------|-----------|
| 4.2.5    | Rework and Follow-up               | 20        |
| 4.2.5.1  | Roles                              | 21        |
| 4.2.5.2  | Outcomes                           | 21        |
| 4.2.6    | Comparison of Activities and Roles | 21        |
| 4.2.7    | Summary of Modifications           | 22        |
| 4.2.8    | Integration Issues                 | 22        |
| 4.3      | Alternative Approaches             | 23        |
| 4.3.1    | Autonomous Activity                | 23        |
| 4.3.2    | Other Formalized Processes         | 23        |
| <b>5</b> | <b>Summary</b>                     | <b>25</b> |
|          | <b>References</b>                  | <b>27</b> |

---

# List of Figures

|   |    |
|---|----|
| Figure 1: Model-Based Verification Activities                         | 1  |
| Figure 2: The Model-Based Verification Paradigm                       | 7  |
| Figure 3: Interplay of the Dimensions                                 | 8  |
| Figure 4: Model Checking  | 11 |
| Figure 5: Four-Step “Process-Formal” Inspection or<br>Review          | 14 |
| Figure 6: Sequential Partitioning of an Artifact                      | 16 |
| Figure 7: Modular Partitioning of an Artifact                         | 16 |
| Figure 8: A Comparison of the Roles of Modeler and<br>Other Reviewers | 18 |
| Figure 9: Four-Step Review Process and Its<br>Outcomes                | 21 |
| Figure 10: Comparison of Responsibilities                             | 22 |





---

# Acknowledgments

The authors would like to thank Chuck Buhman, Chuck Weinstock, and Dave Zubrow for their insightful comments and suggestions on this document. We also would like to acknowledge the contributions of William Thomas in helping us to prepare the final manuscript. Finally, we extend our thanks to the members of the technical staff within the Dependable System Upgrade initiative for their suggestions on defining model-based verification practices.



---

# Abstract

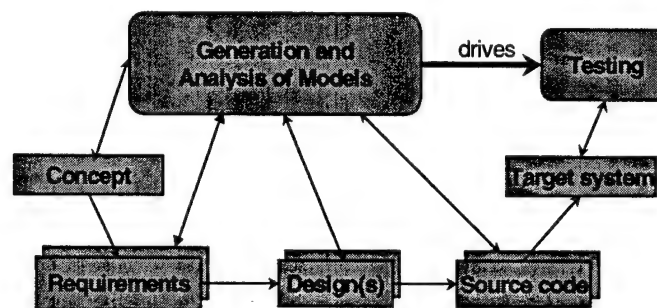
This is an introductory report on the use of model-based verification techniques within software development and upgrade practices. It presents the specific activities and responsibilities that are required of engineers who use the model-based verification paradigm and describes proposed approaches for integrating model-based verification into an organization's software engineering practices. The approaches outlined in this report are preliminary concepts for the integration of model building and analysis techniques into software engineering review and inspection practices. These techniques are presented as both practices within peer review processes and as autonomous engineering investigations. The objective of this report is to provide a starting point for the use of model-based verification techniques and a framework for their evaluation in real-world applications. It is expected that the results of pilot studies that employ the preliminary approaches described here will form the basis for improving the practices themselves and software verification generally.



# 1 Introduction

Model-based verification is a systematic approach to finding and correcting defects (errors) in software requirements, designs, or code [Gluch 98]. While it is based upon formal methodologies, the approach uses formalism to provide a disciplined and logical analysis practice, rather than for “proof” of correctness. This focused application of formalism is used to identify and correct errors in the artifacts of software that is being developed or upgraded and to define testing throughout the life cycle.

The practice of generating formal models early in the development or upgrade of software establishes a systematic software verification practice for understanding a system and identifying errors. The results of these error detection activities can then be used as the basis for testing by defining test strategies, test cases, and critical areas that require focused or more extensive testing. A high-level overview of these activities is shown in Figure 1.



*Figure 1: Model-Based Verification Activities*

Model-based verification relies on formal models to establish a precise way of thinking about the meaning of a software product while providing the foundation for a systematic process for its verification. In particular, the targeted use of formalism in the form of reduced complexity models, or essential models, brings a common and unambiguous language, consistency, precision, and structure into the verification process.

An earlier technical report [Gluch 98] outlined the foundations of model-based verification. This report focuses on the engineering practices associated with the implementation of model-based verification. This report should be considered a preliminary presentation of the concepts associated with the introduction of model-based verification into an organization. The practices described will form the basis for real-world investigations of the techniques. It

is expected that the details of these practices will evolve based on the results of these investigations. Subsequent publications will extend and modify the practices introduced here.

Section 2 discusses the use of models and, in particular, formal essential models in model-based verification. Section 3 describes the responsibilities and activities expected of a software engineer, acting as a modeler within model-based engineering practices. In Section 4, an approach for integrating model-based verification practices into an organization's peer review/inspection processes is presented. The report is summarized in Section 5.

---

## 2 Using Models for Verification

Model-based verification is a two-step practice of model building (creation) and model checking (analysis) for finding errors in software artifacts. These practices combine established software modeling methods with promising techniques emerging from academic and corporate research communities. These practices have also been used successfully in other engineering domains, such as commercial microprocessor design. The choice of a technical approach for a particular situation is based upon the critical—i.e., important or risky—aspects and the type of system being analyzed. Because most of these techniques involve the selective, focused use of formalism, they have been termed *lightweight formal methods* [Jackson 96].

Many of the models that are employed within model-based verification encompass a variety of mathematically based techniques that are not classified as formal methods, while others are founded upon rigorous formal methodologies. Model-based verification also includes models that address the diverse and potentially problematic technical aspects of complex systems. For example, Generalized Rate Monotonic Analysis [Klein 93] models can be employed for the analysis of real-time systems.

### 2.1 Formal Models

In model-based verification, the term model, adapted from Jackson [Jackson 95], refers to a representation that has

- some degree of inherent formalism (logic)
- rules for manipulating components
- important properties shared with the modeled artifact

A simple example that demonstrates these characteristics is the mathematical equation  $E = mc^2$ . This equation is a highly abstracted model of how energy and matter are related. It shows how much energy can be generated from a given amount of mass. It is a formal statement where the mathematics provides a formalism (logic) that prescribes how elements of the model can be manipulated. In this case, the mathematical rules associated with equations enables the re-expression of mass in terms of the energy,  $m = E/c^2$ . This expression shows explicitly the amount of mass that can be formed from a given amount of energy. Note that this is a highly abstracted representation of what happens in a nuclear reaction, focusing only on one aspect of the complex dynamics involved. There are a multitude of other factors that could have been considered (e.g., the momentum spread, or radiation produced).

It is important to note that what makes this a model is the identification of each of the terms of the equation with something in the real world. In this case  $m$  is identified with mass,  $E$  with energy, and  $c$  with the speed of light. This model states that the linear relationship between the symbol  $E$  and the symbol  $m$  that is explicit in the mathematics, is also the same as the relationship between the physical quantities of energy and mass. Not only do the properties of a valid mathematical model mirror the static characteristics of the represented system, but the manipulation of the mathematical symbols associated with that model reflects the behavior of the system as well.

Many of the models used for software engineering are formal and involve a variety of discrete mathematical and logical formalisms. A common model of a software system is a finite state machine model. In finite state models the complex aspects of a system are represented as a finite set of distinct states. For example, the states of a flight control system model may include take-off, cruise, landing, and ground operations.

In contrast to the examples cited above, a simple narrative description of a system or a sketch is not a model that is useful in model-based verification. Generally, software specifications are descriptions written in English prose. The prose, and hence the specification, is only constrained by the rules of syntax and conventional usage. These natural language specifications lack a formal logic that states how to manipulate the elements of the specification.

## 2.2 Essential Models

A central concept in model-based verification is to create simplified models of the critical (important and risky) parts of a system rather than detailed models of the complete system. These simplified models are termed *essential* models. The  $E = mc^2$  model discussed earlier is an essential model of a nuclear reaction, focused on a critical aspect of performance: energy transitions. Thus, essential models are targeted, reduced-complexity portrayals of the system to be verified. They are distinct from other engineering artifacts and are analyzable. By judiciously choosing what parts to model, what perspective(s) to take, and what detail to apply, essential models can provide insight into the critical static and dynamic properties of a system.

The number, level of detail, and formalism of the models, and the view used in model-based verification, are adjusted to meet the particular objectives of the verification activity and the complexity of the system. In some cases only a handful of models that focus on only one or two different perspectives, or only models of minimal formalism, are required. In others, more rigorous formal models would be developed that can be analyzed using automated tools. The range of formalism can extend from using a basic state machine model to employing a complete axiomatic formal system with accompanying syntax, semantics, and inference rules. The foundations and the processes for making these decisions are based upon a core set of engineering principles and formal techniques that can be integrated into existing software engineering review processes.



In describing the integration of model-based verification practices into peer review processes, it is useful to partition peer reviews into two levels of activity:

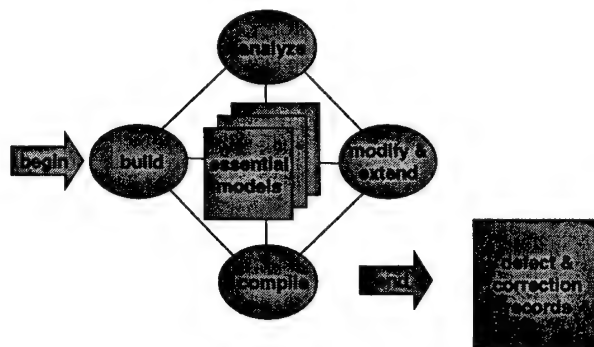
- Individual level. The individual level is made up of the activities of a software engineer participating in the review—the model-based verification paradigm. At this level the details of the engineering decisions and practices required of individual participants are defined.
- Team level. The team level encompasses practices involving multiple individuals, organizations, and teams. It deals with the impact of model-based verification on the procedural aspects of conventional reviews.



### 3 Model-Based Verification Paradigm

The model-based verification paradigm consists of the engineering techniques and practices employed by a software engineer for verifying software requirements, design specifications, or code, through the building and analysis of essential models. Shown as ovals in Figure 2, the activities of the paradigm are

- build; systematically build essential models of the system
- analyze; analyze those models, often using model-checking techniques and tools
- compile; gather detailed information on errors and potential correction actions
- modify and extend; modify and update the essential models



*Figure 2: The Model-Based Verification Paradigm*

Each of these activities has a specific focus and variable duration. Within the paradigm, they are non-sequential, iterative, and interdependent. At the center of all of the activities are essential models. The connections between the activities represent the information exchanged among them and the interdependent control interactions. The control interactions start, suspend, reorder, skip, or terminate an activity. For example, during the modeling process itself, an error may be identified in an artifact. Perhaps an incomplete definition is identified. This situation would require the engineer (modeler) to record the defect, seek clarification, and perhaps modify, reanalyze, extend, or completely replace portions of the model.

A fundamental premise of the paradigm is that building models and analyzing models are complementary activities, where aspects of each result in improving both. For example, through model building a greater understanding of the details of a system is achieved, helping to further focus the analysis on the system's more important or difficult aspects. By analyzing a model developers gain greater insight into its weaknesses and limitations, which helps to guide their subsequent model-building and modification activities. This interplay is facilitated

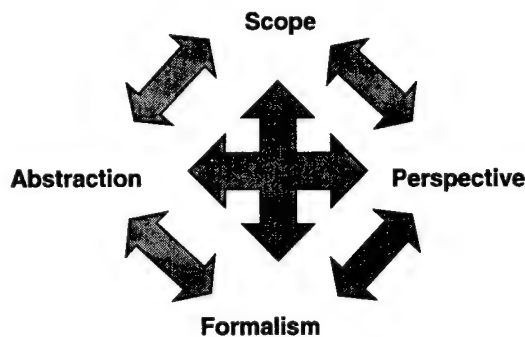
by an iterative process of building a small portion and analyzing a small portion. A small-scale iteration cycle enables a separation of concerns and helps to reduce complexity, simplify analysis, and limit the scope of the impact of mistakes made in the modeling process.

### 3.1 Building Essential Models

Building essential models is a creative engineering enterprise. It involves addressing several perspectives and finding simplified representations that faithfully portray a system. To accomplish this simplification, the model building process involves engineering choices (trade-offs) along four distinct dimensions:

- Scope. What parts of the system need to be modeled?
- Perspective. What modeling views of the system are required?
- Formalism. What formalism should be applied?
- Abstraction. What level of detail is needed?

As shown in Figure 3, these dimensions are not mutually exclusive; rather there is interplay among them. For example, deciding scope and perspective may involve a tradeoff between modeling the whole system from a single perspective or only critical parts of the system from multiple perspectives.



*Figure 3: Interplay of the Dimensions*

The context or nature of the problem domain is particularly influential in the choices among these competing dimensions. This tradeoff process can be viewed as one of optimal matching between domain complexity and model fidelity. In these decisions the four dimensions provide a framework in which to evaluate the alternatives based upon what is important and what is risky, rather than simply providing mutually exclusive choices (i.e., choosing only one particular perspective). In the sections that follow, each of these dimensions is summarized.

### **3.1.1 Scope**

The scope of the model defines how much or what portion of a system is to be modeled. In some cases, it is important to model all of the system at a high level of abstraction considering one perspective (e.g., model and analyze the fault response of the entire system). In other cases it is necessary to model a specific part of the system in greater detail (e.g., model and analyze the communication module). These choices are largely driven by the critical (important and/or risky) aspects of the system and its development, including both programmatic and technical issues. For example, in reviewing the design specification for a networked system, the protocol design is identified as critical. It then becomes the focus of the modeling effort and a formal state machine technique is used to analyze its behavior. In practice these decisions should be viewed as operational ones, such that as the model is built and analyzed, adjustments are made based upon the results of the analysis processes.

### **3.1.2 Perspective**

The modeling perspective is the context for representing the behavior and characteristics of a system. A perspective could be the user's view of a system or it may be a representation of a specific feature, function, or capability of a system. For example, in looking at an aircraft fly-by-wire computer system, one perspective would be to characterize (model) the system from a fault response perspective. This might involve a model that describes states of the system in terms of the number or types of faults that may occur (e.g., the no fault state, single fault state, loss of roll attitude information, etc.). An alternative perspective might be to consider the flight modes of the system (e.g., takeoff, climb out, cruise, landing, etc.).

### **3.1.3 Formalism**

Formalism relates to the level and type of mathematical rigor used for an essential model. The level of mathematical formalism required for effectively modeling a particular system involves decisions that often relate to programmatic as well as a technical issues. A high level of formalism is generally very costly and the potential benefits must be assessed relative to the risks and criticality of a system. While extensive formalism is appropriate for high assurance and safety-critical systems, the cost-effective application of formalism in most system development and upgrade efforts entails an impact assessment of cost and criticality. The choice of the type of formalism results from a consideration of the technical characteristics. Depending upon the nature of an individual problem and the critical aspects of the system, some techniques are more effective and efficient than others. For example, consider a networked financial records management system for general office use. The critical aspects of this system are judged to be data integrity and the complexity associated with multiple concurrent accesses. If there is an error in either, the financial losses may be significant. In this situation, formalized automated model checking would be employed to analyze both data integrity and system concurrency. The analysis of data integrity issues would likely involve mathematical sets and relationships, whereas a state machine model would likely be employed to analyze the system's concurrency.

### 3.1.4 Abstraction

Abstraction is the process of focusing on the important details and ignoring, or abstracting away, those details that are less important. While abstraction is used throughout a development effort, the focus on simplification required for building essential models contrasts with the extensive detail that is needed in the development of requirements, design specifications, or code.

In the development of a system, details and redundancy of representations can be useful, especially if they involve different perspectives. In abstracting essential models, the detailed representations created in the development process must be streamlined into concise, accurate, and less complex models that faithfully represent a system's properties and behavior. Some representative abstraction techniques that can be employed by a modeler include

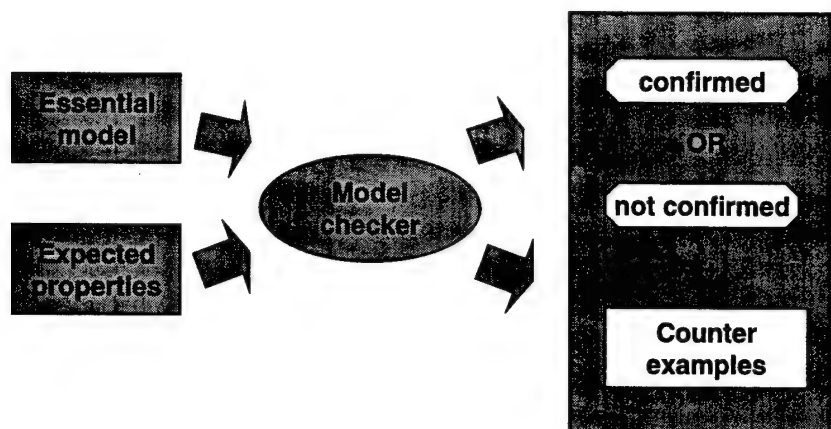
- Systematically decompose the system into components that can be viewed as distinct interacting parallel processes. For example, a large process control system implemented in software on a single processor can be modeled as five separate modules. Each module would have a distinct responsibility.
- Strip away parts of the system that are not relevant to the properties to be demonstrated or proven. For example, it may be the case that variable C depends on variable B and ultimately variable A. If an objective of the model is to understand how variable A affects variable C, it is possible to eliminate variable B from the model by representing the relationship only between A and C.
- Replace a variable with an abstracted or simplified one. This can be viewed as an approach of approximating complex data types with simpler ones. For example, a flight control system specification may contain statements such as "if the airspeed is less than or equal to 200 mph then the gain variable,  $k = 2.1$ ; if the airspeed is between 200 and 400 mph then  $k = 2.3$ , and if the airspeed is equal to or greater than 400 mph then  $k = 2.8$ ." In this case rather than represent the airspeed as a real number, an abstracted representation is used, where airspeed is enumerated as {low, medium, high}.
- Explicitly reduce the size of the system by reducing the number of "equivalent" components. For example, if a distributed networked client-server system has twenty clients and five servers, the system may be modeled as three clients and two servers.
- Use nondeterminism to model unpredictable or unknown inputs to the system. If a model must accept any of a set of events in any order at any time, do not try to model these explicitly. Instead, leave enumeration of possible circumstances to the model checking tool's ability to express nondeterministic behavior.

## 3.2 Analyzing Models (Model Checking)

Analyzing an essential model involves a systematic assessment of its properties and behavior. The analysis criteria are developed from the system requirements, the specification or code, its antecedent specifications, and the involvement of domain experts. This involvement can include interviews with users, customers, or domain engineers or their direct participation in defining these criteria. Analysis criteria are often expressed as expected properties of the system, are focused upon the critical aspects of the system, and include domain specific and technical considerations. For example, consider the analysis of an essential model of a weap-

ons launch control system. The analysis would likely include confirming that a weapon would be launched when ready and commanded to do so and could not be launched if it is not ready. In addition, technically detailed criteria like the absence of starvation, livelocks, and deadlocks in the software design would be included in the analysis.

While manual analysis of models can provide insight into many aspects of the system, a more effective approach, especially for complex or concurrent systems, is to employ automated analysis, or model checking [Clarke 96]. Figure 4 shows the model-checking process where an essential model of a system is “checked” against the analysis criteria, expressed as expected properties. The output of the model checker is either a confirmation of the validity of each expected property or a non-confirmation. In most cases if the property is not confirmed by the model checker, a counter example is provided.



*Figure 4: Model Checking*

Model-checking approaches include a variety of techniques and tools [Clarke 96]. Most model checkers (model-checking tools) require the essential model to be a finite state machine. During automated analysis, a model checker “explores” a model’s state space to determine the validity of the expected properties relative to the model.

One of the major challenges associated with model checkers is the state explosion problem—the extraordinarily large number of states that result from complicated models. The size of this state space can number in the hundreds of billions of states, exceeding the storage capacity of even the largest computers and requiring decades of computation time. Thus, essential models are vital for analyzing software systems using model checkers. In building essential models, it is important to recognize both the capabilities and limitations of model checking.

A second and equally important challenge in model checking involves expressing the expected properties of the model. The language for stating expected properties is dependent upon the particular model checker. Many state machine model checkers use temporal logic statements. While temporal logic itself is not difficult to understand, temporal logic expressions can be arcane. Working with them requires a sound understanding of the idiosyncrasies

of the particular temporal logic representation. For example in the Symbolic Model Checker (SMV), a property that *any request for service will result in that request eventually being acknowledged* is expressed as:

```
AG(request-> AF acknowledged).
```

### 3.3 Compiling Defects

Compiling defects consists of recording the defects that are identified both in the artifact being reviewed and in the models themselves, and integrating these defects into the larger set associated with the overall verification process. As noted, this is an activity that pervades all aspects of the paradigm. During the building and extending of models, defects in the artifact are uncovered and recorded. Throughout the analysis, the identified defects and related counter examples are included in the set of defect logs for the system.

No specific procedure or defect recording format is required for the model-based verification paradigm. A general standard format or one unique to an organization can be employed. What is critical is that the results are integrated into the overall verification process for the project and the larger organization.

### 3.4 Modifying and Extending Models

As defects and uncertainties are identified, their resolution may require additional or alternative modeling activities. The increased insight into the system resulting from modeling activities also provides a basis for more focused modeling efforts. For example, consider the model-based verification of a client-server networked database system. In modeling the client-to-server message exchange protocol, subtle interdependencies among individual components within the client software are uncovered. This realization is then used as the foundation for exploring the interdependencies throughout the entire system.



---

## 4 Model-Based Verification in Peer Review Processes

In general, it is possible to integrate model-based verification practices into existing processes. This section outlines an approach for integrating model-based verification practices into peer reviews. It is believed that with minor modifications this approach can be used with most peer review styles.

### 4.1 Reviews and Formalism

Peer reviews of software requirements and design specifications have been shown to be effective in detecting errors in the early phases of software development [Wheeler 96]. Many of these are informal (e.g., design or code walkthroughs). Others are more formal (e.g., formal inspections), where the term formal refers to administrative and procedural aspects, not to a foundation of mathematical formalism. These “process-formal” reviews involve checklists (general and specialized), defined roles and responsibilities, well defined outputs, process metrics, and expected behaviors [Fagan 76, Fagan 86, Humphrey 97, Gilb 93, STR 96, Wheeler 96]. A notable exception to these reviews are the mathematically formal verification reviews used in the cleanroom method [Mills 87].

The purpose of walkthroughs, peer reviews, and inspections is to improve the quality of the artifact under review by removing defects. A fundamental premise of peer reviews is that others will find defects that the creator of the artifact overlooked. Review of the artifact by someone other than its creator can involve an individual or a group of peers.

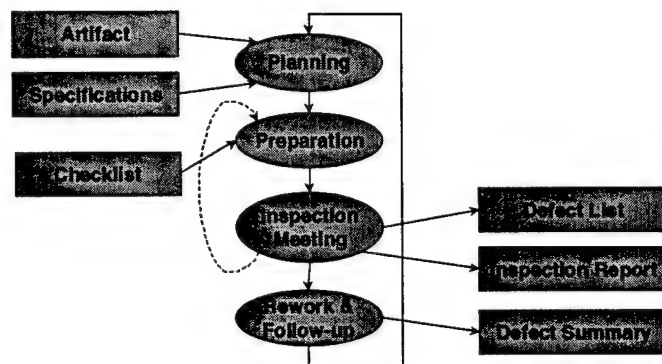
Through facilitated group interaction processes, it is believed that more defects will be uncovered than through reviews by a single reviewer or by multiple individual reviewers whose efforts are not coordinated. Even with several individuals working together on a problem, the effectiveness of peer review processes is limited by a human’s ability to analyze complex problems. Model-based verification attempts to improve a review team’s ability to deal within complexity by supplementing the “process formality” of facilitated group interactions with selective mathematical formality—formal modeling practices.

### 4.2 Integration of Processes

In this section, a four-step generic peer review process is used as a framework for discussing the integration of model-based verification practices into a peer review process. This generic process is shown in Figure 5. It represents a “process-formal” review that is similar to the

inspection processes proposed by a number of authors (e.g., the IEEE standard for software reviews and audits (1028-1988) [IEEE, Gilb 93, Fagan 76, Wheeler 96]. The overall process is abstracted into the four principal process steps of planning, preparation, meetings, and rework and follow-up. In this representation it is assumed that there may be multiple cycles of preparation and meetings, as shown by the dotted arrow in Figure 5. The multiple cycles are to ensure that a large artifact is reviewed in smaller partitions. An important result of this partitioning is that an inspection (group) meeting is no more than two hours long.

The major inputs to the process are the artifact to be reviewed—requirements or design specifications—or code, checklist(s), guidelines, and related materials for use in the review process. The general set of outputs of a review include detailed and summary listing of the defects, a report on the results of the inspection with process metrics, and related data.



**Figure 5: Four-Step "Process-Formal" Inspection or Review**

The process depicted in Figure 5 is a team activity where each participant assumes a particular role and associated set of responsibilities as defined by the specific method. For example, one participant assumes the role of scribe in the Fagan method of inspection. The inclusion of model-based verification practices introduces a new role, a modeling specialist (modeler). A modeler is expert in the application of a variety of modeling techniques. There may be one or more members of the team that are assuming this role. This addition of the modeler into the peer review process requires that the overall process accommodate the new role. In particular, this addition requires that other inspectors coordinate (both in timing and substance) their activities with those of the modeler. This coordination is the dominant consideration in the integration of model-based verification practices.

While the specifics of an implementation of this style of peer review will vary, the four-step characterization of the process outlined above will be used as a framework to discuss the integration of model-based verification practices into peer review processes. In the next four sections the potential impact of the use of model-based verification techniques in each of the major steps is addressed. The focus of each section is on the effects that model building and analysis have on the processes, rather than a detailed explanation of the activities involved in

each step. These impacts may relate directly to modeling and analysis or to modifying checklists and other elements of the conventional review process.

### 4.2.1 Planning

The initial step in conventional reviews consists of planning activities that define the objectives, approaches, and plans for execution of the review. When including model-based verification practices in reviews, outcomes of the initial planning step should include

- which aspects of the system should be modeled
- which modeling techniques are appropriate
- what properties of the system must be verified

Critical aspects of the system should be used as the bases for these decisions. The amount of risk involved as well as the importance of relevant requirements determine which aspects are critical. Since generally it is not feasible to model all of the system in detail, the choice of these critical aspects requires substantial domain knowledge as well as knowledge about the relevant implementation details of the system. To be effective in these decision processes it is imperative that either the leader of the review team or the modeler—and preferably both—have this broad understanding of the system's requirements, design, and environment. If this is not the case, a knowledgeable individual with this perspective should be included in the planning activities.

Knowledge of the capabilities of the various modeling techniques is vital to making the right choice. In general, decisions on specific modeling techniques should include considerations of the characteristics of the system being modeled, the efficacy of the modeling technique when applied to those characteristics, the complexity of the model being generated, and the risks associated with the system. In particular, the risks can help to determine the level of formality at which a system should be analyzed (e.g., a highly formalized model would be appropriate for a safety-critical system). High assurance often implies high cost, however, and these types of tradeoffs should be considered when choosing a modeling technique.

The planning activity should also produce a preliminary set of key expected properties to be used in the analysis of the essential models of the system. This preliminary set is one that should evolve throughout the review process, incorporating modifications as additional insight into the system is developed. The expected properties as well as the results of the modeling effort can also be used to modify checklists that are used by the other reviewers.

The decisions on scope and techniques are interleaved with other factors, including the level of formality, perspective of the modeling effort, and abstraction levels that may be required. The techniques on how to address these engineering decisions is part of the Model-Based Verification Paradigm that is described in Section 3. The issues relating to the risky aspects often involve knowledge of development issues (e.g., awareness that a particular model

turned out to be far more difficult to design than originally planned) as well as formal software engineering expertise.

#### 4.2.1.1 Methods of Partitioning

In a review of a complex software system, it is often necessary to partition the artifact that is being reviewed. This partitioning is accomplished to ensure that each part can be effectively reviewed within a two-hour time limit for an inspection meeting. The method used to partition the artifact can affect how model-based verification is integrated into the process.

Two possible partitioning approaches, as shown in Figure 6 and Figure 7, are sequential and modular. Sequential partitioning divides the artifact into sections according to its physical organization—by page number for example. Modular partitioning breaks the work product into general conceptual areas, or modules, that reflect the system's structure (e.g., design modules or functional areas in a requirements specification).

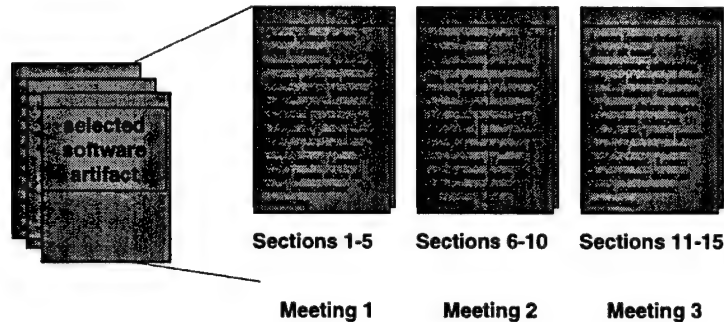


Figure 6: Sequential Partitioning of an Artifact

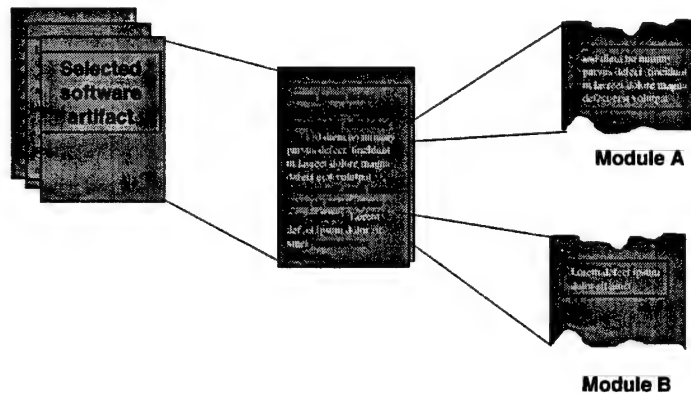


Figure 7: Modular Partitioning of an Artifact

## 4.2.2 Coordination

It is important that the activities of the modeler and those of the other reviewers are coordinated. Coordination allows the modeler to participate in the inspection meetings. The modeler benefits from coordination by gaining insight into the details of the system. The other reviewers benefit by seeing the additional defects identified through the modeler's activities. All benefit from cross-checking the work done by the others. In general a modular partitioning can facilitate this coordination, but through appropriate planning, the sequential method can also be structured to enable effective coordination of all of the reviewers.

In structuring the plan for the review, timing issues also affect the coordination activities. Planning for adequate time during the preparation steps and between meetings ensures that meaningful modeling and analysis can be completed. The input of the modeler and the use of historical metrics on the time and resources required to complete modeling and analysis efforts are vital for developing a sound plan for the review. Currently the data on modeling in software systems is limited [Srinivasan 98] but as model-based verification practices mature the quantity and quality of these data will improve.

### 4.2.2.1 Impact on Roles

The modeler's role in the planning stage is more extensive than that of other reviewers. Both the modeler and the other reviewers must become familiar with the work product. In addition, the modeler must participate in the planning activity by contributing to the decisions on partitioning the artifact, establishing the scope, defining the techniques, and developing the basic plan for building and analyzing the essential models of the system.

The integration of model-based techniques into the planning activity also requires the review team leader to have some familiarity with model-based verification practices. Detailed knowledge of how to employ a specific technique on the part of the leader is not required. The leader must understand the general capabilities and limitations of each modeling technique. This knowledge will allow the leader to contribute to decisions on the choices of scope and modeling technique.

### 4.2.2.2 Outcomes

In summary, the outcomes of the planning step that are affected by the integration of model-based verification techniques are

- **Review plan.** This is a normal outcome of a review process. When model-based verification techniques are employed in the review process, the plan includes modifications that reflect the additional outcomes and that integrate the activities of the modeler with the rest of the review team.
- **Modeling plan.** This plan establishes the modeling-specific activities and captures the results of the decisions on modeling scope, techniques, and perspective. In addition, it includes a description of the expected properties that will be used in the analyses of the models.

### 4.2.3 Preparation

The preparation stage is the time when individual reviewers, working alone, analyze the artifact as they seek to understand the system and identify defects. In doing so, they rely on procedures, the responsibilities of their assigned role, and generally, a checklist or guideline as the basis for identifying defects. While also working independently, modelers are focusing on the building of models as the basis for analyzing the artifact under review and, as appropriate, analyzing those models manually or with model-checking tools. The difference between the activities of a modeler and a reviewer is depicted in Figure 8.

A reviewer in most conventional reviews relies on a checklist or guidelines to facilitate the identification of defects. In contrast, modelers rely on the rigor and formalism required of a particular modeling technique to guide and facilitate their efforts in uncovering defects during the building of the models. As shown in Figure 8 both the analysis activity of a modeler, as well as the process of building the models result in the identification of defects. In building a model, the need imposed by the formalism to specify exactly and completely the range of legal values for a variable can uncover hard-to-detect errors in a specification, such as the use of *greater than* ( $>$ ) rather than *greater than or equal to* ( $\geq$ ). The value of simply building a model was demonstrated in a recent research project. This project involved assessing the use of modeling and model checking techniques in the analysis of the Traffic Alert and Collision Avoidance System (TCAS II). As the model was being built from the requirements specification, an error was found. This error involved the use of *greater than* ( $>$ ) rather than *less than or equal to* ( $\leq$ ) [Anderson 96]. During the analysis process (model checking), the expected properties of the system are used as the basis to check the models. Depending on the particular model-checking technique used, automated analysis efforts can provide guidance in correcting the defects, while also identifying defects.

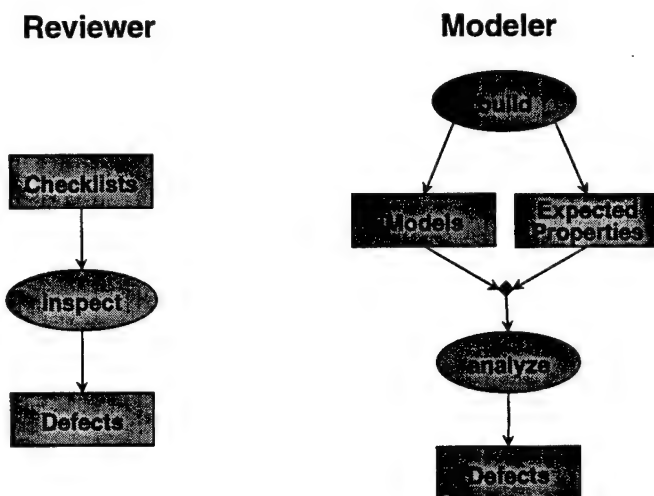


Figure 8: A Comparison of the Roles of Modeler and Other Reviewers

#### **4.2.3.1 Impact on Roles**

The modeler's role in the preparation stage has the same purpose as that of a reviewer—finding defects. But while the other inspectors are looking for defects by visually inspecting the work product, the modeler builds and analyzes models to assess the expected properties of the system identified in the planning stage. The activities associated with the other reviewer roles are generally not affected by the work of the modeler.

#### **4.2.3.2 Outcomes**

At the end of the preparation stage, the modeler should have completed and analyzed a model or set of models that cover the essential aspects of the system as determined in the planning stage. These models, analyses, and any defects identified are presented and logged at the review meeting along with the defects found by the other reviewers.

#### **4.2.4 Meeting**

In the meeting step, also known as the logging meeting, all review participants meet to

- record defects found during preparation
- look for additional defects
- present the system models and any defects exposed by the analyses
- determine areas of future work
- address suggestions and questions about the process

Generally, the leader begins the meeting by giving a brief overview of the material that will be addressed during the meeting and then the reviewers report on defects found in their individual preparation activities. Various protocols are used in this meeting, and the interactions and associated discussions that occur can uncover new defects. As an outcome of this meeting the defects identified individually and as a team are recorded.

##### **4.2.4.1 Coordination**

Coordinating the efforts of the modeler and those of the other reviewers is critical to ensuring effective review meetings. The modeling time estimates, timing for the meetings, and the interactions among the reviewers must be defined clearly during the planning stage. These plans may include the allocation of a longer time for the modeler to complete the modeling and analysis effort to enable the modeler to participate in a particular review meeting. Alternatively, the review plan may involve limited participation by the modeler in one or more meetings. For example, if there are five meetings planned to review an artifact, a modeler may attend one or more meetings with limited involvement. This may include simply noting defects identified by others or presenting the model as it currently stands. In other cases a modeler may not attend all of the meetings but rather spend the meeting time modeling and analyzing the artifact. In these circumstances a modeler would benefit by reviewing the results of the meeting, including the defects captured to aid in the modeling effort.



#### **4.2.4.2 Modeler Role**

During the meeting, modelers would participate much as any other reviewer in that they would present, classify, and record defects. In presenting defects, the modeler would rely on the results of the modeling and analysis completed to that point. The modeler can use errors recorded during the meeting to guide subsequent modeling and analysis efforts. These errors can help identify critical areas of the system that may be high risk or have a high error occurrence rate, thereby further focusing the modeling and analysis work. In addition, by working with the review team leader the modeler can use the results of the modeling and analysis efforts to modify the checklists for the continuing review of the artifact.

One additional but optional responsibility of a modeler is the presentation of the models developed during the modeling process. In doing so, the modeler presents the system model, describing its scope, its perspective, and the expected properties used in model checking. While this presentation is optional, its purpose is to identify errors in the model, errors in the assumptions made by the modeler, or deficiencies in the analysis. The defects identified in the model during this presentation should be noted and used to modify and focus future modeling and analysis efforts.

A key aspect of presenting the model or an analysis of a model to the team is the level of formality of the presentation. In particular, the modeler should not assume a knowledge of formal methods on the part of the other reviewers. To accomplish this, the modeler should rely on the abstractions employed in the model rather than focusing on details of the model or modeling language. Many model-checking tools provide a counterexample when an error is found, but these are often cryptic. Using a domain-specific scenario can help explain the steps that led to the error by relating them to a real-world example.

#### **4.2.4.3 Outcomes**

The outcome of the meeting step is principally a record of the defects identified by all of the participants throughout the process—a defect log. As part of this log or as an additional artifact, the errors and issues that involve the models are also captured. This additional information regarding the models is used in subsequent review activities and in the rework and follow-up step.

#### **4.2.5 Rework and Follow-up**

This last step in the process is a closure activity that involves completing the rework and confirming its completion. The rework involves making changes to the artifact based upon the correction of defects and modifications of the models that may be required. Since the model is designed to mirror critical aspects of the artifact, it must be kept consistent with changes to the artifact. Thus, for any rework done on the artifact, it is necessary to determine the impact on the models. If the models are impacted by the rework, they should be revised to stay consistent with the artifact, and then reanalyzed. By analyzing the reworked models the team can identify new defects that may have been introduced in the rework.



#### 4.2.5.1 Roles

The modeler has the responsibility to rework models and to re-run model analyses as needed. In addition to the responsibilities normally assumed by the team lead in this step, the lead also must ensure that the rework on the models is completed.

#### 4.2.5.2 Outcomes

At the completion of the peer review process, there are additional outcomes associated with model-based verification practices. These include the models themselves and their related artifacts as shown in Figure 9. This information becomes part of the collective set of artifacts that should be maintained during a development or upgrade effort. They are especially useful in providing a high-level basis for subsequent upgrades.

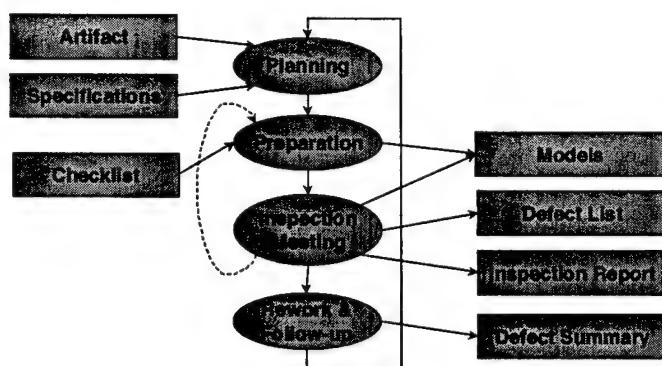


Figure 9: Four-Step Review Process and Its Outcomes

#### 4.2.6 Comparison of Activities and Roles

In integrating model-based verification techniques into conventional reviews, as represented by the generic four-step process, modifications to the process focus on coordinating the activities of the modeler with those of other members of the review team. Figure 10 summarizes relevant activities and responsibilities and contrasts the role of the reviewers who are following the conventional process with that of the modeler. The modeler follows a separate but parallel track that involves explicit coordination in the timing and content of activities at both the planning and meeting steps. In the planning step, this includes working with the team leader and others in planning the modeling activities and coordination efforts. In the review meeting step, this involves a team effort to compile the results of the review processes.

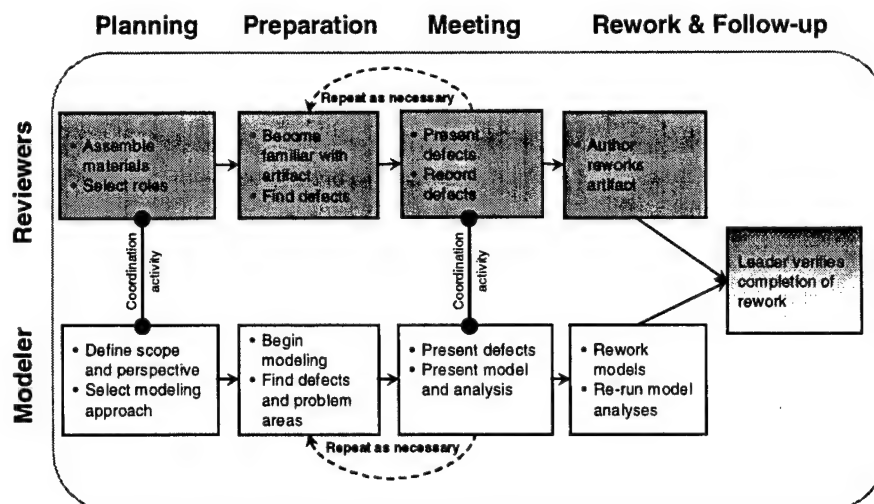


Figure 10: Comparison of Responsibilities

### 4.2.7 Summary of Modifications

The modifications to the generic process required to incorporate model-based verification techniques are summarized in this section. These modifications include the following steps:

- **Planning step.** The modeling process is included in the plan for the review. In particular, decisions must be made on what should be modeled, what modeling techniques to employ, and what expected properties are to be verified. The other considerations include choosing a partitioning approach for the artifact. This choice, as well as considerations of the time and resources required for the review effort, will help the team coordinate modeling activities with those of the general review process.
- **Preparation step.** Individual reviewers analyze the artifact under review while modelers investigate errors by building and analyzing models of the artifact. Modelers rely on the formal framework of the modeling technique rather than a checklist or other guide.
- **Meeting step.** Modelers participate much as the other reviewers in the process do, presenting defects in the group setting. The defects noted can be used to guide subsequent modeling and analysis efforts. Optionally, the modeler may present the models to the larger group for its review and comment.
- **Follow-up and rework step.** The modeler reworks models and reruns analyses as required. The key is to ensure that the models accurately reflect the revised artifact.

### 4.2.8 Integration Issues

Integrating model-based verification practices into existing review processes comes with some potential difficulties. The modeler's responsibilities in the early stages of review, planning, and preparation can take longer than other reviewers' tasks. This is especially true when a new modeling technique is chosen or the review artifact is complex. Preliminary data show that model building often takes more time than it takes to prepare for a conventional review,

and is also less predictable. Scheduling should account for this and provide the modeler with sufficient time to build and analyze models.

The concept of combining model-based verification and peer review is still untested. It is hoped that future pilot studies will provide data with which to compare this conceptual review process with conventional peer reviews.

## **4.3 Alternative Approaches**

A generic four step peer review process was used as a basis for demonstrating how model-based verification techniques can be integrated into most peer inspection and review methods. This section discusses alternative ways to integrate model-based verification concepts and practices into an organization or project. In all of these approaches mathematical formalism is intended to supplement the process formality of inspections and other established peer review processes.

### **4.3.1 Autonomous Activity**

It is possible to have an autonomous model-based verification activity that involves the use of individual reviewers who are experts in the model-based technique but are not part of a team review process. These autonomous reviews may be employed periodically to analyze a complete system, to address a specific issue, or to focus on a particularly important technical aspect or requirement involved in the development or upgrade of a system. These activities would be extensions to existing practices, not substitutes for established peer review processes. These styles of review may be particularly effective for unanticipated events or as part of a risk mitigation strategy.

### **4.3.2 Other Formalized Processes**

Formalism is an integral part of the cleanroom method and is a vital component of the verification procedures used in that method [Mills 87]. Recently other review approaches, which are representative of the model-based verification philosophy of relying on formalism to bring a systematic structure to the review process, have been proposed and investigated. For example, a comprehensive formal review technique has been defined that includes mathematically formal tabular representations [Parnas 85, Parnas 94]. Also, there are review processes that attempt to focus the work of individual reviewers relying upon the engineering aspects of the system [Porter 95]. In a recent article Johnson [Johnson 98] discusses reengineering inspection processes, including alternative approaches to enhance the effectiveness and acceptance of process formal reviews.



---

## 5 Summary

Creating and analyzing essential models is a challenging engineering enterprise. The keys to its effectiveness are

- incremental and iterative modeling techniques
- focus on critical system aspects
- integration with existing processes
- appropriate use of human reviewers and model-checking tools

The model-based verification paradigm relies heavily on the modeler. This person must be familiar with both system requirements and modeling techniques. Other members of the peer review group can assist the modeler by offering different perspectives on system requirements, helping the modeler determine an appropriate scope and level of abstraction to use when creating essential models.

Cooperation between the modeler and other reviewers is important. Model-checking tools can manage complexity that human reviewers cannot, and human activities such as abstraction and risk assessment help direct the modeling effort. A process that combines model-based verification with traditional peer reviews leverages the strengths of human reviewers and model-checking tools by using each appropriately. The strengths of human reviewers are the ability to

- find different perspectives from which to examine a system
- abstract critical aspects from irrelevant detail
- identify important system properties

These talents help direct the modeler in creating and analyzing models. A model-based verification effort will involve a set of models of various scopes, perspectives, and levels of abstractions. Creating and analyzing essential models supplement conventional peer review by providing

- discipline enforced by mathematical formalism
- explicit mathematical rigor
- thorough examination of all possible interactions
- conclusive proof by example when a defect is found

The approaches outlined in this report are preliminary and will be the basis for pilot studies using model-based verification in real-world development efforts. This work represents an incremental advance in evolving the practice. Establishing guidelines and principles for the decisions on what to model and to what level is one of the major areas of investigation of the model-based verification work. Results of these studies will be used to augment the practices described here and will form the foundation for more detailed guides for the practice of model-based verification.

---

## References

- [Anderson 96] Anderson, R. J.; Beam, P.; Burns, S.; Chan, W.; Modugno, F.; Notkin, D.; & Reese, J. D. "Model Checking Large Software Specifications," 156-166. *SIGSOFT '96, Proceedings of the Fourth ACM Symposium on the Foundations of Software Engineering*. October 1996.
- [Clarke 96] Clarke, E. M. & Wing, Jeannette. "Formal Methods: State of the Art and Future Directions." *ACM Computing Surveys* 28, 4 (December 1996): 626-643.
- [Fagan 76] Fagan, M. "Design and Code Inspections to Reduce Errors in Program Development." *IBM Systems Journal* 15, 3 (1976): 182-211.
- [Fagan 86] Fagan, M. E., "Advances in Software Inspections." *IEEE Transactions on Software Engineering SE-12*, 7 (July 1986): 744-751.
- [Gilb 93] Gilb, T. & Graham, D. *Software Inspection*. Wokingham, England: Addison-Wesley, 1993.
- [Gluch 98] Gluch, D. & Weinstock, C. *Model-Based Verification: A Technology for Dependable System Upgrade* (CMU/SEI-98-TR-009, ADA354756). Pittsburgh, Pa.: Software Engineering Institute, Carnegie Mellon University, 1998.
- [Humphrey 97] Humphrey, W. S. *Introduction to the Personal Software Process*. SEI Series in Software Engineering, Reading, Mass.: Addison Wesley Longman, Inc., 1997.
- [IEEE 89] ANSI/IEEE Std. 1028-1988. *Standard for Software Reviews and Audits*. Software Engineering Standards, Third Edition, 1989.
- [Jackson 95] Jackson, M. *Software Requirements and Specifications: A Lexicon of Practice, Principles, and Prejudices*. New York: ACM Press, Addison-Wesley, 1995.

- [Jackson 96] Jackson, Daniel & Wing, Jeannette. "Lightweight Formal Methods." *IEEE Computer* (April 1996): 21-22.
- [Johnson 98] Johnson, P. M. "Reengineering Inspection." *Communications of the ACM* 41, 2 (February 1998): 49-52.
- [Klein 93] Klein, M.; et al. *A Practitioner's Handbook for Real-Time Analysis: Guide to Rate Monotonic Analysis for Real-Time Systems*. Boston, Mass.: Kluwer Academic Publishers, 1993.
- [Mills 87] Mills, H.; Dyer, M.; & Linger, R. "Cleanroom Software Engineering." *IEEE Software* 4, 5 (1987):1987.
- [Parnas 85] Parnas, D. L. & Weiss, D. M. "Active Design Reviews: Principles and Practices." 132-136. *Proceedings of the Eighth International Conference on Software Engineering*, August 1985.
- [Parnas 94] Parnas, D. L. "Inspection of Safety-Critical Software Using Program-Function Tables," 270-277. *Proceedings of IFIP 13<sup>th</sup> World Computer Congress, Vol. A-53*, Hamburg, Germany, Aug. 28-Sept. 2, 1994. IFIP Transactions A (Computer Science and Technology).
- [Porter 95] Porter, A. A.; Votta, L. G. Jr.; & Basili, V. R. "Comparing Detection Methods for Software Requirements Inspections: A Replicated Experiment." *IEEE Transactions on Software Engineering* 21, 6, (June 1995): 563-575.
- [Srinivasan 98] Srinivasan, Grama R. & Gluch, D. P. *A Study of Practice Issues in Model-Based Verification Using the Symbolic Model Verifier (SMV)* (CMU/SEI-98-TR-013, ADA358751). Pittsburgh, Pa.: Software Engineering Institute, Carnegie Mellon University, 1998.
- [STR 96] "Software Inspections." *Software Technology Review* [online]. Available WWW: <[http://www.sei.cmu.edu/str/descriptions/inspections\\_body.html](http://www.sei.cmu.edu/str/descriptions/inspections_body.html)>.
- [Wheeler 96] Wheeler, D. A.; Brykczynski, B.; & Meeson, R. N. Jr. *Software Inspection: An Industry Best Practice*. Los Alamitos, Calif.: IEEE Computer Society Press, 1996.



# REPORT DOCUMENTATION PAGE

Form Approved  
OMB No. 0704-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.

|  |   |  |   |
|--|---|--|---|
| <b>1. AGENCY USE ONLY (LEAVE BLANK)</b>  |   | <b>2. REPORT DATE</b><br>April 1999                            | <b>3. REPORT TYPE AND DATES COVERED</b><br>Final                      |
| <b>4. TITLE AND SUBTITLE</b><br>An Introduction to Software Engineering Practices Using Model-Based Verification   |   |  | <b>5. FUNDING NUMBERS</b><br>C — F19628-95-C-0003                     |
| <b>6. AUTHOR(S)</b><br>David P. Gluch<br>Jared Brockway  |   |  |   |
| <b>7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)</b><br>Software Engineering Institute<br>Carnegie Mellon University<br>Pittsburgh, PA 15213  |   |  | <b>8. PERFORMING ORGANIZATION REPORT NUMBER</b><br>CMU/SEI-99-TR-005  |
| <b>9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)</b><br>HQ ESC/DIB<br>5 Eglin Street<br>Hanscom AFB, MA 01731-2116   |   |  | <b>9. SPONSORING/MONITORING AGENCY REPORT NUMBER</b><br>ESC-TR-99-005 |
| <b>11. SUPPLEMENTARY NOTES</b>   |   |  |   |
| <b>12.A DISTRIBUTION/AVAILABILITY STATEMENT</b><br>Unclassified/Unlimited, DTIC, NTIS  |   |  | <b>12.B DISTRIBUTION CODE</b>   |
| <b>13. ABSTRACT (MAXIMUM 200 WORDS)</b><br><br>This is an introductory report on the use of model-based verification techniques within software development and upgrade practices. It presents the specific activities and responsibilities that are required of engineers who use the model-based verification paradigm and describes proposed approaches for integrating model-based verification into an organization's software engineering practices. The approaches outlined in this report are preliminary concepts for the integration of model building and analysis techniques into software engineering review and inspection practices. These techniques are presented as both practices within peer review processes and as autonomous engineering investigations. The objective of this report is to provide a starting point for the use of model-based verification techniques and a framework for their evaluation in real-world applications. It is expected that the results of pilot studies that employ the preliminary approaches described here will form the basis for improving the practices themselves and software verification generally. |   |  |   |
| <b>14. SUBJECT TERMS</b><br>model-based verification, model checking, verification, formal reviews, formal inspections, software engineering practices   |   |  | <b>15. NUMBER OF PAGES</b><br>40                                      |
|  |   |  | <b>16. PRICE CODE</b>   |
| <b>17. SECURITY CLASSIFICATION OF REPORT</b><br>UNCLASSIFIED   | <b>18. SECURITY CLASSIFICATION OF THIS PAGE</b><br>UNCLASSIFIED | <b>19. SECURITY CLASSIFICATION OF ABSTRACT</b><br>UNCLASSIFIED | <b>20. LIMITATION OF ABSTRACT</b><br>UL                               |

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)  
Prescribed by ANSI Std. Z39-18  
298-102